Formal Methods in Conformance Testing: Result and Perspectives

Ana R. Cavalli[a], Jean-Philippe Favreau[b], Marc Phalippou[c]

[a]Institut National des Télécommunications, Les Epinettes, 9 rue Charles Fourier, 91011 Evry Cedex, France, Email: Ana.Cavalli@int-evry.fr

[b]Computer Systems Laboratory, National Institute of Standards and Technology, Bldg. 225, B218, Gaithersburg, MD 20899, USA, E-mail: favreau@osi.ncsl.nist.gov

[c]France Telecom-CNET LAA/SLC/EVP, Route de Trégastel, BP 40, F-22301 Lannion Cedex, France, E-mail: Marc.Phalippou@lannion.cnet.fr

Abstract

The application of formal methods to conformance testing becomes a more and more active research area. This paper presents the results and perspectives of the application of these languages for tests and test generation methods. These results are analyzed in the framework of the activity of the joint ISO/ITU-TS working group on "Formal Methods for Conformance Testing."

Keywords: Conformance Testing, Formal Description Techniques, Testing Theory, Standardization.

1. INTRODUCTION

With the aim of providing open markets and allowing competition between equipment manufacturers, international agreements on testing methods and procedures become a priority. Such agreed methods should simultaneously provide a very good technical basis, in order to guarantee the optimal efficiency of the testing process, and result from a general consensus in order to achieve world-wide mutual recognition of the test results.

The ISO 9646 standard [ISO91] provides a methodology and framework applicable to the conformance testing of OSI products. This standard has been mainly oriented towards practical needs. It incorporates a great deal of practical experience issued from test experts which have been involved in concrete test issues. However, formalism in this standard is limited to definition of the TTCN language as a test notation. This language is now widely used in the telecommunications area for the description of test suites.

The increasing use of formal description techniques such as Estelle [ISO89A], LOTOS [ISO89b] and SDL [CCITT] for the specification of telecommunications systems, as well as new developments in the theory of testing, should undoubtedly have a significant influence on the evolution of testing technology. This is why the International Standards Organization (ISO) and International Telecommunication Union - Telecommunication Standardization Sector (ITU-TS) started a joint project on "Formal methods in conformance testing" (FMCT) with the following objectives:

1. Investigate how the progress of testing theory allows a better understanding of the ISO 9646 concepts. For instance, what is the formal interpretation of conformance, of test suite, of test execution, of test coverage? Specifically, how can we apply these concepts to specification expressed in the above mentioned formal description techniques?

2. Evaluate the possibility to improve the test development process through an automatic or computer-aided test case generation from formal specifications. For this, a precise evaluation of the state-of-the-art is needed.

3. Provide guidelines to write testable specifications expressed in the three standardized formal description techniques.

This paper[1] presents a summary of the work which has been performed up to now by the FMCT working group. After an overview of the main research directions which are related the work of the group, we analyze through some examples their concrete influence on the standardized draft document. We show that some concepts from the informal framework cannot be directly applied to formal specifications, and that new insights on some problems can be obtained from a formal approach. Finally, we conclude with an analysis of the perspectives that are offered now for the evolution of this work.

2. STATE OF THE ART AND CURRENT STATUS OF FMCT WORKING DRAFT

It is not the purpose of this paper to present an exhaustive overview of the current research in such a wide area as conformance testing. In this part, we will concentrate on an analysis of the main research directions which have a direct influence on the work of the FMCT working group. Two questions are relevant:

1. The *formalization of testing concepts*. It is the subject of the main body of the FMCT working draft document, because it is important to standardize a common understanding of such basic concepts to reach a mutual recognition of test results.

2. The *test generation methods*. The FMCT group has chosen a practical approach to evaluate several test generation methods, based on the study of a common example, INRES protocol [Inr 93]. Since it did not appear relevant to the FMCT group to standardize a particular test generation method, the latter study is, included as Annex B of the document.

We present here a summary of the approaches that have been investigated.

2.1. Formalization of Testing

One of the goals of the FMCT working group is to "establish a theory and framework...which may be sued to assess conformance of an implementation to behavior specified in a formal description." This means that the following concepts must be formalized:

1. what does conformance mean?
2. what does testing mean?
3. how can we model the testing procedures, and in particular the verdict assignment?

Few theories have been proposed to answer these questions. The most detailed work on this subject has been performed at Twente University, and is reported for instance in [Tret 91]. It has been chosen as a basis for the test of the FMCT working draft.

Starting from ISO 9646 concepts, *conformance* is interpreted as the satisfaction of a set of conformance requirements, which are derived from the specification of a system. Given a specification expressed in a formal way

[1]This paper presents part of the work which is performed in the ISO/ITU-TS group on "Formal Methods for Conformance Testing." However, it is a synthesis which only represents the point of view of the authors, and should not be considered as an official presentation of the group's work. This paper is a contribution of the Institut National des Télécommunications, France Telecom-CNET LAA/SLC/EVP and the National Institute of Standards and Technology and is not subject to copyright.

(one of the three standardized FDTs for instance), the set of conformance requirements defined by the specification can be obtained through a particular relation. The conformance requirements are expressed in a suitable formalism, which may be different from the specification formalism, for instance temporal logic.

These requirements are a set of properties that all conforming implementations should satisfy. If these properties are expressed in a logical formalism, they constitute a theory, and the conforming implementations are interpreted as the models which satisfy these theories. This is why we call this a logical model of conformance. In this approach the implementations may, but are not restricted to, be described also by a standard formal description technique.

Unfortunately the standardized FDTs do not directly describe the conformance requirements that conforming implementations should satisfy. Instead, Estelle, LOTOS and SDL provide descriptions of the behavior of the specification, in terms of the observable events that the specified systems may offer at their external interface (in a black box approach to specification). In this case, it is not easy, and not necessarily useful, to describe explicitly the conformance requirements. The conforming implementations can be directly related to the specification by means of an *implementation relation:* given a set of potential implementations (for instance described in the same FDT as the specification), this relation indicates which ones are conforming to a given specification. We call this approach a behavioral mode of conformance. It is more adapted to study conformance to specifications which are expressed in FDTs. It seems to be different from the concept of conformance used in ISO 9646, but equivalence between the logical and the behavioral modeling of conformance has been proved, and therefore both representations can be interchanged.

Testing is a way to assess conformance of an implementation to its specification by means of a test experiment. In practice, a test suite is composed of a finite number of test cases. Each one of these test cases is designed to check some particular conformance requirements, which are called the test purpose. In order to know the result of one test case, several test runs (or executions) may be necessary. There exists several theories which intend to capture all these concepts. Some of them, like [Abr 87] adopt a denotational approach of testing. Only the final verdict of a test case is modeled here. The way to assign a verdict is represented by a function which maps the set of test cases to the set of verdicts. No attempt is made to give some more intuitive "operational" modeling of test execution.

In some other approaches [Bri 88], testing is represented by the parallel composition of a tester with the implementation under test. The tester is an abstract model of a whole test suite. In this theory, no interpretation is given to individual test executions, only the global verdict is interpreted. Moreover, the verdict is expressed as a global property of the system composed of the implementation in parallel with the tester, but no attempt is made to describe how to obtain this verdict in an operational way (i.e., by means of an algorithm). The same model of testing as a parallel execution is refined in [Tre 91] [Tre 92] to take into account individual test cases.

In the classical test methods based on finite state automata [Cho 78], [Nai 81], [Sab 88], [Vuo 89] and [Fuj 91a], no explicit modeling of the test activity is done. Conformance is assumed to be trace equivalence between automata, and the testing process is an exchange of interactions between a tester and the implementation under test. Since in the above mentioned papers specifications and implementations are always completely specified and deterministic automata, there is no difficulty in assigning a verdict to the test. Recently, the need to take into account more complex models, where non-determinism and incompleteness play an important role, highlighted the need for other implementation relations, and for a more sophisticated approach to verdict assignment.

However, for the time being, these new developments of the testing theory are not mature enough to be included in the FMCT working draft. Therefore only a very general formalization of conformance testing, based on the concept of observations, which model the result of test executions, is provided. Further contributions are expected on this subject as the state-of-the-art evolves.

An important point in testing is the choice of a particular *test architecture*. This choice is often limited in practice, due to the particular constraints of the real test environment. For instance, some products do not offer an accessible interface to each layer of an OSI protocol stack, but only an access to the application layer through the user interface. In other cases, when remote testing is performed, the upper interface of the implementation under test may not be available for testing, and each test is performed only through the lower interface. Theoretical results have shown that the limitation on controllability and observability of implementations, due to the presence of a test context (or interface) between the tester and the implementation, have a significant influence on what kind of requirements can be checked by testing [Pha 92] [Ver 92].

Automatic test generation is one of the major benefits that may be expected from a formal approach to conformance testing. Although it is not among the objectives of the FMCT group to standardize a particular test generation method, a precise evaluation of the state-of-the-art in this domain is important. We summarize in the following sections some approaches which are studied within the FMCT group.

2.2. Test Generation Methods Based on Testers

The testing theory developed in [Bri 88] includes a test generation part. This theory is developed for specifications which are expressed in LOTOS. A particular implementation relation, named *Conf*, is studied. The choice of this particular implementation relation is guided by technical reasons, too detailed to be included in this paper. Testing an implementation is modeled as the parallel composition of a LOTOS process, called a tester, with another LOTOS process modeling the implementation under test. The test verdict is based on the existence of deadlocks in the system composed of the tester and the implementation.

[Bri 88] shows that it is possible to define for every specification a particular tester, named the *canonical tester*, which characterizes the conforming implementations of the specification. That means that the canonical tester, when composed with any possible implementation, will conclude to a success if and only if the implementation conformance to the specification.

Unfortunately, this very interesting result cannot be taken into account when concrete test generation for real protocols is attempted. Although in [Wez 90] an algorithm is defined to compute the canonical tester, this tester has an infinite behavior as soon as the specification has an infinite behavior. Moreover, the success is defined as a global property of a system which, due to the non-determinism of the implementations, may show several different behaviors when executed. No operational procedure is given to decide concretely success or failure.

Due to these severe limitations, no experiment of test generation based on the canonical tester has yet been performed within the FMCT group.

2.3. Test Generation Methods Based on Checking Experiments

In this section we will present some of the more representative methods designed to derive tests from Formal Descriptions (FDs) of a protocol based on checking experiments. These methods have their origins in the *checking experiment problem* from automata theory, in which the objective is to determine experimentally whether a given state table describes the behavior of a FSM implementation.

This approach tries to give an efficient answer to the following major issues that occur when performing a test:

1. Bringing the IUT (Implementation Under Test) into a desired state (or condition) starting from an initial state (or condition);

2. Applying the necessary stimuli and observing the output (including null output);

3. Verifying that the IUT is in the expected state (or condition);

4. Bringing the IUT into an initial state.

Issues 1 and 4 are concerned with *controllability of testing*, while issues 2 and 3 concern *observability*. Issues 1 and 4 correspond to preambles and postambles of ISO 9646, respectively, and 2 and 3 to the test body.

Some methods derive tests directly from the FD. Other methods consist of a transformation of the specification written in a standardized Formal Description Technique (FDT) -- Estelle, LOTOS or SDL -- into a model described in the same or another formalism and subsequently obtaining test generation from this model. This transformation may either preserve all information in the FD, resulting in an intermediate model that is equivalent to the FD, or not preserve all the information, resulting in an *intermediate model* that retains only specific aspects of the FD. In both cases it can be also considered as an abstract model of the implementation.

Currently the following *intermediate models* for test generations are considered: Finite State Machines (FSMs), Input/Output Finite State Machines (I/O FSMs), Extended Finite State Machines (EFSMs), Labeled Transition Systems (LTSs), Asynchronous Communication Trees (ACTs) and Charts.

The transformations applied to the specifications in order to obtain an intermediate model and also in order to reduce this intermediate model can be characterized as *implementation relations*. In fact, these transformations permit to obtain a model of the implementation. Examples of implementation relations are *trace equivalence* [Hop 79], *trace pre-order*, *failure equivalence*[Bro 84] and *Conf* relation [Bri 88].

In the presentation of these methods the following aspects will be emphasized: the implementation relations and the methods used to derive conformance requirements and test purposes. The limitations on error-detection and the concrete application of each technique will also be presented.

The method of Transitions Tours [Nai 81], [Boc 82] is based on a I/O FSM and tries to find a tour that traverses each graph edge at least once. The implementation relation is trace equivalence and all conformance requirements are deduced from the specification. It allows checking the output function of every transition and to obtain an executable finite test sequence. This method only addresses the controllability issue and doesn't address the observability issue.

Other methods to solve the controllability problem have been proposed. The Distinguishing Sequences (DS) method [Hen 64] [Koh 78] [Gon 70] tries to find an input sequence for the I/O FSM such that the output sequence generated is distinct for each starting state. The implementation relation for this method is trace equivalence, all conformance requirements are deduced from the specification. It also permits obtaining a finite tester, to test every transition, check-output and transfer functions. This method only addresses the observability issue and presents as drawbacks that not every I/O FSM has a DS and that they are very long (upper bound is (n-1)n for an n-state I/O FSM).

The Unique Input/Out [Sab 88] sequences tries to find an input sequence for each state such that the output sequence generated is unique to that state. The UIO method is similar to DS concerning implementation relations, conformance requirements, obtaining a finite tester and the test of output and transfer functions. It also requires the number of states in the implementation to be the same as in the specification, the I/O FSM must be deterministic, minimal and possess the reset capability. The difference with DS method is that more I/O FSMs possess an UIO sequence than a DS sequence and that this method addresses the observability issue but also the controllability issue when combined with the rural Chinese Postman tour [Aho 88].

The concept of UIO sequence has been adapted to the Finite Labeled Transition graph obtained from LOTOS specifications [Cav 92a]. The Unique Event (UE) sequence tries to obtain an unique event sequence for each state such that it is only accepted by this state. It can also be combined with the rural Chinese Postman tour to solve both problems of controllability and observability.

The W-method [Cho 78] tries to find a set of input sequences for each state such that the output sequence generated is unique to that state. These methods have similar characteristics, as the DS and the UIO methods, concerning implementation relations, conformance requirements and obtaining a finite test sequence. The advantage is that all I/O FSMs possess a W set and the drawback is that it is typically very long.

Methods as DS, UIO, UE have as a drawback that not all I/O FSMs possess this kind of sequence for each state. Different solutions have been proposed to solve this problem: signatures [Sab 88] defined as an unique Input/Output sequence that differentiates each state from every other. This definition of signature fails to generate a correct test sequence when used in conjunction with an optimization algorithm. To solve this problem other solutions have been proposed as partial UE sequences [Cav 93].

Another problem is the uniqueness of UIO sequences and signatures in the implementation. A method that integrates this verification has been proposed [Vuo 89], it is also applied to the verification of the uniqueness of signatures.

Improvements to obtain shortest test sequences have been proposed using techniques based on the overlapping of test sequences [Sid 89] and the elimination of the reset requirement [Yao 93].

As a conclusion of this section, it must be mentioned that some of these methods have been applied to the INRES protocol and the results of this experience are presented in the annex B of the FMCT working draft document.

The experience shows that these methods can be characterized because:

1. They are fully automated methods. They provide automated generation, from the formal description of the protocol, of test cases and optimized test sequences
2. They are able to process the INRES example, but they have some problems with the size of the intermediate model (I/O FSMs, finite LTSs) used for test generation.
3. They have pragmatic objectives (automating and optimizing) but they are based on theoretical approaches (automata theory, equivalence relations). They allow checking implementation relations as trace pre-order and trace equivalence and they can be adapted to check failure equivalence and *Conf* relations.

2.4. Test Generation Methods Based on Asynchronous Trees

These methods are based on the construction of a tree that represents the behavior of the system. The root is the initial system state, the nodes are all the other system states and the arcs represent the valid transitions. This method that has been applied mainly to the SDL language tries to generate a complete but finite tree even from a set of non-terminating SDL processes. The arcs of the tree define the complete conformance test suite. However, a method must be defined for dealing with non-determinism [Brö 89, Bou 89].

These methods can be automated, but the main problem is that the tree may sometimes be infinite making very difficult the task of finding criteria for test selection. They don't provide yet what kind of implementations relations are checked.

2.5. Computer-Aided Test Generation Methods

Apart from the above methods which are based on checking experiments theory for finite state automata, we have also experienced within the FMCT group some more empirical approaches to test generation.

TVEDA [Pha 90] generates test cases from an extended finite state machine (EFSM). An EFSM is an automaton (states and input/output transitions) extended by data. Data are used as variables, for refining the structure of the states, or as parameters for refining the structure of inputs and outputs. The firing of transitions, and the inputs and outputs which are manipulated depend on the value of data. The main feature of such an EFSM, when compared to classical finite state automata, is to allow a compact description of automata: the use of data allows a kind of factorization of the transitions.

The strategy TVEDA used for generating tests for one EFSM has been inspired from the test strategy used manually for low layer protocols such as ISDN LAPD protocol: one test is created for each extended transition. Starting from the initial state of the EFSM, a preamble brings the IUT into the start state of the extended transition (step 1). Then the transition is activated, the output from the IUT is checked (step 2), and a check sequence checks that the final state of the IUT is correct (step 3). The current status of the tool is that only the parts of test cases corresponding to the tested extended transition (step 2) is generated: the preamble and the check sequence are not produced by TVEDA. This is why we call this a computer-aided test generation, not a fully automated one.

The FMCT group has also studied another computer-aided test generation method, which takes as input an SDL specification, and some test purposes expressed in message sequence charts [Ell 93]. This method is supported by a tool called STED.

Some attempts have also been developed to transform the three FDTs (Estelle, LOTOS and SDL) into a common semantic model [TR189]. The common semantic model has the form of an EFSM called Chart. Although Chart based models may be well adapted to represent each FDT separately, a major problem of this method is the mutually exclusive fields for LOTOS on the one hand, and SDL or Estelle on the other hand. If the EFSM-chart requires different interpretations for SDL or Estelle, and LOTOS, then it is not a common semantic model.

When compared to the methods which are presented in the previous section, these methods can be distinguished because:

1. They are only computer-aided methods, not fully automated. TVEDA method computes only test skeletons from a formal specification, not complete test cases. STED needs the additional description of test purposes in order to compute the test cases

2. They are able to process easily the INRES example. In both cases the number of test cases which are produced is reasonable, and corresponds to the expectations of human test experts. Results are analyzed in the annex B of the FMCT working draft
3. They are empirical in the sense that they have been designed with the objective of providing a reasonable help to test developers. On the contrary, it is not yet possible to relate such methods to a formal theory of testing. In particular, it seems quite difficult to formulate what kind of implementation relation is checked by the test suites which are computed with these tools!

3. INFLUENCE OF FMCT WORK ON TESTING METHODOLOGY

In the previous section we have reviewed the main directions for which formal methods have been applied to the problems of conformance testing, at least in the framework of the FMCT working group (we know that this covers only very partially the research activity in this field, but this is due to the limited number of contributions that have been received, and everyone who has got some results which are not mentioned above is welcome in the FMCT group).

We show now, in this section, that the above studies based on formal methods can provide new insights on some problems which cannot be completely analyzed in an informal setting. We do not claim to be exhaustive here. We have selected four examples which illustrate the variety of results that can be obtained.

3.1. Importance of Implementation Relations

The first example we chose is not directly related to testing methodology, but is rather a consequence of testing experience on the use of formal specification languages.

The formalization of testing has shown in effect that *current formal specifications are of no use* if they do not come together with a precise definition of the implementation relation which is considered. In effect, a formal specification should represent an abstract description of the future implementations to be realized. Every specification should therefore characterize precisely which implementations correspond to the specification. This is specially important for testing, because there can be no notion of conformance if this is not provided.

The choice of an implementation relation is arbitrary. It is a convention that holds between the specifier and the tester. Examples have been given to illustrate the wide choice of potentially interesting implementation relations [Tre 92]. However, we must insist on the fact that none of the three standardized FDTs provides a "standard" implementation relation. In this sense, and to be a bit provocative, we can say that a formal specification alone specifies nothing precise.

Let us take a concrete example based on the INRES protocol. It is not the purpose of this paper to describe in detail once again this well-known connection-oriented data transfer protocol. What is important here is to know that this protocol makes use of well-defined service primitives, and that the precise description of the protocol indicates how these primitives can be chained to provide the INRES service. Let us notice that such a general statement applies to most of the known protocols, which show the representativity of INRES. When testing this protocol, the tester expects to find that the allowed chaining of service primitives is respected by the conforming implementation.

Now let us suppose that we have chosen a trace inclusion as our implementation relation, i.e., "every sequence which is shown by the implementation should be allowed by the specification". Note that this is not so stupid, and is often heard. Then an implementation which behaves like a deaf-mute, refusing to accept messages and never answering to any primitive (i.e. an implementation with an empty set of traces) is a correct implementation.

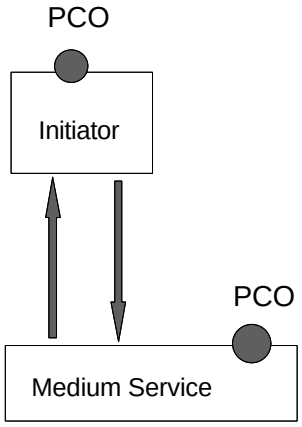
On the contrary, suppose that we chose the reverse trace inclusion as our implementation relation, i.e. "every sequence which is in the specification should be provided by the implementation". Here again this is not so stupid, and is often heard. Then an implementation which behaves like a game of chance, which answers in a non-deterministic way any of the defined primitives (i.e. an implementation which has all possible traces) is a correct implementation.

This example should convince the reader of the need for a reasonable implementation relation.

3.2. Influence of Test Architecture

Theoretical studies on the power of testing have shown that the test architecture limits the controllability and observability of testing, and that these limitations have a strong influence on the conformance requirements that can be effectively checked by means of testing. Depending on the test architecture, some conformance requirements may be no longer testable.

Here again let us take an example based on the INRES protocol. This example is a bit extreme, but it is designed to illustrate the effects of what we call the test context (i.e. the interface between the tester and the implementation under test). Assume that we are using a remote test method, and therefore that the lower tester has access to the implementation under test through a channel, as described in the following picture:



This test architecture has been chosen within the FMCT group for the experiment on INRES. Let us assume that the channel is very unreliable, and that it can lose, reorder or create arbitrarily messages. Then:

1. If we test an incorrect implementation which has a correct behavior at the upper interface, but an incorrect one at the lower interface (for instance, this implementation sends no message at all at the lower interface, due to a faulty hardware component. Clearly this is a non-conforming implementation). Unfortunately, it will be impossible to detect that it is a faulty implementation, for the messages may be lost by the channel, and we cannot conclude that the fault comes from the INRES implementation. If the channel can create messages, we may even receive the expected messages, in spite of the fact that the implementation is incorrect.

2. On the contrary, if we test a correct implementation (for instance an implementation which performs exactly as the specification requires, nothing less, nothing more), and if we apply to this implementation a standard manually generated test suite, then we have reasonable chances to reach incorrect fail verdicts. For instance, after having received ICONreq from the user, INRES is supposed to send a CR at the lower interface. Therefore the test suite is likely to have a test that checks this behavior, and this test will emit a fail verdict if nothing is detected by the lower tester within a reasonable delay. In our case, if the message is lost by the channel, the fail verdict will occur, in spite of the fact that the implementation is correct.

This analysis of the effects of the test interface on the testability of conformance requirements leads to two conclusions:

1. On the design of specifications. Testing constraints should be taken into account during the specification phase. It is useless to specify behavior that will not be testable in any case due to test constraints. In particular, the design of the test interfaces of the system should be done carefully, ensuring that every conformance requirement is testable at these test interfaces. If the protocol that we want to test is supposed

to be included in a protocol stack for instance, then it is useless to specify requirements which will not be observable through the stack.

2. On the design of test suites. As we explained above, some test cases which are correct with a given test architecture may no longer be correct with another test architecture (they may give wrong fail verdicts or wrong pass verdicts). Therefore it is obvious that the test architecture should be taken into account during the development of the test suite. The traditional ISO 9646 procedure to design test suites (first test purposes, then generic test cases, then abstract test cases) should probably be modified. Abstract test cases cannot be expressed as a refinement of generic test cases (i.e. test cases applicable with an ideal test architecture, which would provide optimal observability and controllability), neither as a selection among these generic test cases. Our point of view, in the light of theoretical results, is that abstract test cases should be directly derived from the test purposes and the test architecture. [Note that "Generic" has been dropped from 9646].

3.3. Chaining of Individual Test Cases

The problem we describe in this section has been presented in a contribution to a recent FMCT meeting, and has been detected during test generation experiments with INRES protocol [Bau 93].

Assume that the INRES implementation is tested through a remote test architecture, i.e. through a channel, as described in the figure of the previous section. Assume that the channel may introduce a delay in the transmission of messages, and that this delay is not well-known in advance, which is a plausible hypothesis. Then, due to the specified behavior of the INRES protocol, an unknown number of old CR repetitions (caused by the time-out mechanism) may be left in the medium at the end of one particular test case. These may turn up at the beginning of a new test case, maybe in coincidence with a new connection, possibly leading to an erroneous verdict of the new test case.

The only solution that could be found was the introduction of a reset mechanism in the channel, in order to clean the channel between individual test case. A detailed analysis of the problem (taking into account the TTCN semantics for Point of Control and Observation -- PCO -- queues) has shown that this reset should be acknowledged in order to insure test cases separation. That means that at the beginning of a new test case, there is an emission of a MRESETreq (medium reset request), and a waiting for a MRESETcon (medium reset confirmation). Then the test case can proceed as scheduled.

Let us notice that this solution can be adopted in practice only if the available channel offers such a reset mechanism, and that, as a consequence, all test cases must be modified, with the introduction of the reset interactions.

This leads to the conclusion that the notion of individual test cases which can be manipulated as independent pieces of testing has a meaning only if it is possible to reach a clean situation after the execution of each test case. We should be aware of this additional constraint imposed by a ISO 9646 approach to test suite structure. Notice that this problem does not appear if we use the approach of test generation methods based on checking experiments (see section 2.3), in which a test suite is composed of only one big sequence of exchanges (inputs and outputs) between the tester and the IUT.

3.4. How to Manage Synchronous Specifications

The use of formal methods also puts in evidence some problems about the relationship between synchronous and asynchronous specifications. More precisely, the work developed for the generation of test sequences from the LOTOS specification of a protocol and the translation of these sequences into TTCN illustrates this situation [Lou 93].

Some work has been developed in order to translate these test sequences into TTCN. But LOTOS possesses a synchronous communication by rendez-vous, while TTCN possesses asynchronous communication via PCOs modeled by means of two FIFO queues.

The first solution proposed was to integrate into the LOTOS specification a context composed of two FIFO queues to describe the communication of the system with the environment. This solution was not satisfactory. The LOTOS specification of a simplified version of the Alternating Bit protocol with two FIFO queues of two places produced a too large FSM, as is shown in the following table:

	Without FIFOs	With FIFO
Number of States	6	248
Number of Transitions	8	652

A second solution was adopted: the use of an implicit synchronous/asynchronous interface between the IUT and the tester. The main problem was to express in TTCN the test of a LOTOS rendez-vous that has the form <gate><data>. For example, a LOTOS rendez-vous could be described as SDT!1. The solution was to describe the beginning of the rendez-vous by a TTCN message, and to activate a timer. If the rendez-vous is accepted, the tester will receive a TTCN message corresponding to the end of the rendez-vous. Otherwise, the timer will expire, which corresponds to the rendez-vous being rejected.

3.5. Relations Between the Service and the Protocol

Before the conformance test generation for the INRES protocol, some work to achieve the verification of this protocol has been accomplished.

The specification of the service and the protocol was provided in the three standardized FDTs (Estelle, LOTOS and SDL), but only in the case of LOTOS specifications was it possible to verify that the service and the protocol were equivalent.

In the case of LOTOS, the comparison between the service and the protocol with respect to observational equivalence was possible. Moreover, the verification procedure shows that some situations expected in the protocol were not reflected in the service [Cav 92b].

4. PERSPECTIVES

The work developed by the FMCT group will be oriented towards the improvement of the solutions given to the problems mentioned in the previous sections and the aperture towards new subjects and new problems yet unsolved, but clearly identified. In the following, a non exhaustive list of these old/new problems is given:

- *Non-determinism of specifications*. Specifications in the three standardized FDTs allow non-determinism. This non-determinism is due to the arrival of signals that produce transitions to different states but also due to internal events. How to solve this problem is one of the research subjects that is recently being studied. Different solutions have been proposed [Fuj 91b], but it is still an open problem and research on it is still going on.

- *Testability of specifications*. The use of an FDT does not automatically ensure that the described system is testable. System requirements defined using formal description techniques should be expressed in a way such that they are testable. Specification styles can be defined to provide guidelines to be applied during the specification phase to increase the testability of the resulting specifications. Specification styles for the three standardized FDTs (Estelle, LOTOS and SDL) would be realized.

- *Testability of SDL*. Work on the testability of the SDL language has been already developed [Ell 92]. In the case of this language, for instance, different semantic characteristics affect its testability: the asynchronous communication, the time model and the non-determinism of specifications. Research work has been developed to insure that SDL specifications are testable.

- *Interoperability Testing*. Since conformance testing cannot insure a complete conformity of the implementations to be tested, tests for interoperability become a necessary complement to conformance testing. It will be necessary to prove, for instance, that two implementations that conform to the specification can inter-operate [Raf 90]. Laboratories and industrial constructors have developed methods and tools to test interoperability of protocols, but no work to standardize these tests has been developed.

- *Outside OSI* . The definition of conformance testing for systems not possessing the OSI architecture. When developing Open Distributed Processing (ODP) component standards, it will be crucial to develop at the same time conformance test suites and procedures. This subject includes also the definition of conformance testing for more complex systems as telecommunications systems (PABX, future services) and program interfaces.

Also to be mentioned the need to increase contributions to the FMCT group of poorly treated subjects, as for example:

- *Fault models and test coverage metrics*. Fault models provide the basis for the definition of fault coverage measures and for the design of testing and test selection methodologies. A related problem is how to measure the "goodness" of a set of test cases and how to generate or select test suites with some good coverage measure. Although some research work is being developed on both subjects, [Boc 91] and [Vuo 91], the results of this research has not yet been integrated in the work developed by the FMCT group.

5. CONCLUSIONS

This paper has presented the current status of the work of the ISO/ITU-TS working group on "Formal methods for conformance testing". We have shown that the progress of testing theory and of test generation methods have already allowed to reach a good understanding, and a general agreement within the working group, on the relationship between formal description techniques and the related testing concept.

We have then analyzed the first concrete benefits that have been obtained form the use of formal methods in the analysis of test problems. Clearly these methods allow to gain new insights on some testing concepts, and, in the future, this should undoubtedly have some influence on the evolution of the testing methodology.

Finally, we have underlined that many problems remain open, in particular to extend the scope of the current studies to more general telecommunication systems and services, as requested by ITU-TS objectives. This will be possible only if advanced research teams in this domain provide an increased contribution effort, for which we call here again.

REFERENCES

[Abr 87] S. Abramsky, Observation equivalence as a testing equivalence, Theoretical computer science, vol. 53, pp. 225-241, 1987.

[Aho 89] A. V. Aho, A. T. Dahbura, D. Lee, and U. Uyar, An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, Protocol Specification, Testing, and Verification VIII, S. Aggarwal and K. Sabnani editors, North-Holland, 1988.

[Bau 93] B. Baumgarten, The CAB protocol, contribution to the FMCT meeting, Evry, March 1993.

[Boc 82] G. v. Bochmann and B. Sarikaya, Some experience with Sequence Generation for Protocols, Proceedings Second International Workshop on Protocol Specification, Verification and Testing, North-Holland, Amsterdam, 1982.

[Boc 91] G. v. Bochmann, A. Das, R. Dssouli, M. Dubuc, A. Ghedamsi and G. Luo, Fault Models in Testing, Proceedings of the 4th International Workshop on Protocol Test Systems, Leidschendam, October 1991.

[Bou 89] A. Bourguet-Rouger and P. Combes, Exhaustive Validation and Test Generation in ELVIS, SDL'89: The language at work, O. Faergemand and M. Marques editors, North-Holland, 1989

[Bri 88] E. Brinksma, A theory for the derivation of tests, Protocol Specification, Testing, and Verification VIII, S. Aggarwal and K. Sabnani editors, North-Holland, 1988.

[Brö 89] L. Brömstrup and D. Hogrefe, TESDL: Experience with Generating Test Cases from SDL Specifications, SDL'89: The language at work, O. Faergemand and M. Marques editors, North-Holland, 1989.

[Bro 84] S. D. Brookes, C.A.R. Hoare and A.W. Roscoe, A Theory of Communicating Sequential Processes, JACM 31 (1984), 560-599.

[Cav92a] A.R. Cavalli, S.U. Kim and P. Maigron, Automated Protocol Conformance Test Generation Based on Formal Methods for LOTOS Specifications, Protocol Test Systems, V (C-11), G.v. Bochmann, R. Dssouli and A. Das (Editors), Elsevier Science Publishers B.V., North Holland, 1992.

[Cav92b] A. R. Cavalli and P. Hiong, Generation of a Finite State Machine from a LOTOS Specification of INRES protocol using CAESAR, contribution to the FMCT meeting, Geneva, November 1993

[Cav 93] A.R. Cavalli, S.U. Kim and P. Maigron, Improving Conformance Testing for LOTOS, Rapport de recherche N· 93-05-01, Institut National des Télécommunications, France, May 1993.

[CCITT] CCITT, Specification and Description Language (SDL), Recommendation Z.100, 1988.

[Cho 78] T.S. Chow, Testing Software Design Modeled by Finite State Machines, IEEE Transactions on Software Engineering, Vol. SE-4, Nr.3, S.178-187, 1978.

[Ell 92] Ellsberger J. and Kristoffersen F., Testability in the context of SDL, Protocol Specification, Testing, and Verification XII, Lake Buena Vista, Florida, June 1992.

[Ell 93] J. Ellsberger, Computer supported test generation from SDL specifications, contribution to FMCT working draft, 1993.

[Fuj 91a] S. Fujiwara and G. v. Bochmann, F. Khendek, M. Amalou, A. Ghedamsi, Test Selection Based on Finite State Models, IEEE Transactions on Software Engineering, vol. 17, n. 6, juin 1991.

[Fuj 91b] S. Fujiwara and G. v. Bochmann, Testing non-deterministic state machines with fault coverage, Proceedings of the 4th International Workshop on Protocol Test Systems, Leidschendam, October 1991.

[Hen 64] F.C. Hennie, Fault-detecting experiments for sequential circuits, in Proceeding 5th Ann. Symp. on Switching Circuit Theory and Logical Design, November, 1964.

[Hop 79] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[Inr 93] OSI formal specification case study: the INRES protocol and service, Revised version, Annex E, FMCT working draft, June 1993.

[ISO89a] ISO, Information Processing Systems, Open Systems Interconnection,. Estelle - A Formal Description Technique based on an Extended State Transition Model, IS-9074, 1989.

[ISO89b] ISO, Information Processing Systems, Open Systems Interconnection, LOTOS - A Formal Description Technique based on the Temporal Ordering of Observational Behavior. IS-8807. 1989.

[ISO91] ISO, Information Technology, Open Systems Interconnection, OSI Conformance Testing Methodology and Framework, ISO DP 9646.

[Koh 78] Z. Kohavi, Switching and Finite Automata Theory, New York, McGraw-Hill, 1978.

[Lou 93] S. Loui, Génération automatique de séquences de tests de conformité et traduction en TTCN, Rapport INT, June 1993, France.

[Nai 81] S. Naito, M. Tsunonyama, Fault-Detection for Sequential Machines by Transitions Tours, Proceedings of IEEE Fault Tolerant Computing Conference, S. 238-243, 1981.

[Pha 90] M. Phalippou, R. Groz, Evaluation of an empirical approach for computer-aided test case generation, Proceedings of the 3rd International Workshop on Protocol Test Systems, pp. 131-147, Washington, October 1990.

[Pha 92] M. Phalippou, The limited power of testing, proceedings of the 5th International Workshop on Protocol Test Systems, Montreal, September 1992.

[Raf 90] O. Rafiq and R. Castanet, From Conformance Testing to Interoperability Testing, Proceedings of the 3rd International Workshop on Protocol Test Systems, pp. 371-385, Washington, October 1990.

[Sab 88] K. Sabnani and A. Dahbura, A Protocol Test Generation Procedure, Computer Networks and ISDN Systems, Vol. 15, No. 4, pp. 285-297, 1988.

[Sid 89] D. Sidhu and T. Leung, Formal Methods for Conformance Testing: A detailed study, IEEE Transactions on Software Engineering, SE-15: 413-427, 1989.

[Tre 91] J. Tretmans, P. Kars, E. Brinksma, Protocol conformance testing: a formal perspective on ISO IS-9646, Proceedings of the 4th International Workshop on Protocol Test Systems, Leidschendam, 1991.

[Tre 92] J. Tretmans, A formal approach to conformance testing, Ph.D. thesis, Twente university, 1992.

[Ver 92] L. Verhaard, J. Tretmans, P. Kars, E. Brinksma, On asynchronous testing, proceedings of the 5th International Workshop on Protocol Test Systems, Montreal, September 1992.

[Vuo 89] S. Vuong, W. Chan, M. Ito, The UIOv-Method for Protocol Test Sequence Generation, proceedings of the second International Workshop on Protocol Test Systems, Berlin, October 1989.

[Vuo 91] S. Vuong, Test coverage metrics for protocols, Proceedings of the 4th International Workshop on Protocol Test Systems, Leidschendam, October 1991.

[Wez 90] C. Wezeman, S. Batley, J. Lynch, Formal methods to assist conformance testing: a case study, proceedings of FORTE 90, Madrid, November 1990.

[Yao 93] M. Yao, A. Petrenko and G.v. Bochmann, Conformance testing of Protocol Machines without Reset, Protocol Specification, Testing, and Verification XIII, Liege, Belgium, May 1993.